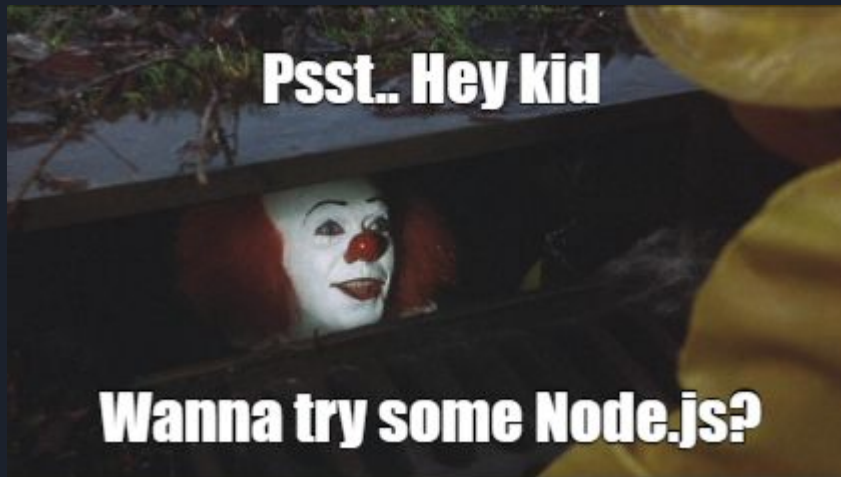


NODEJS

Tarun Singh
10xtarun@gmail.com



Code and PPT link : https://github.com/10xtarun/sttp_node_2021

Pre - Questions

1. What is JavaScript ?
2. Have you ever used JavaScript outside the Browser ?



not the JS you are looking for



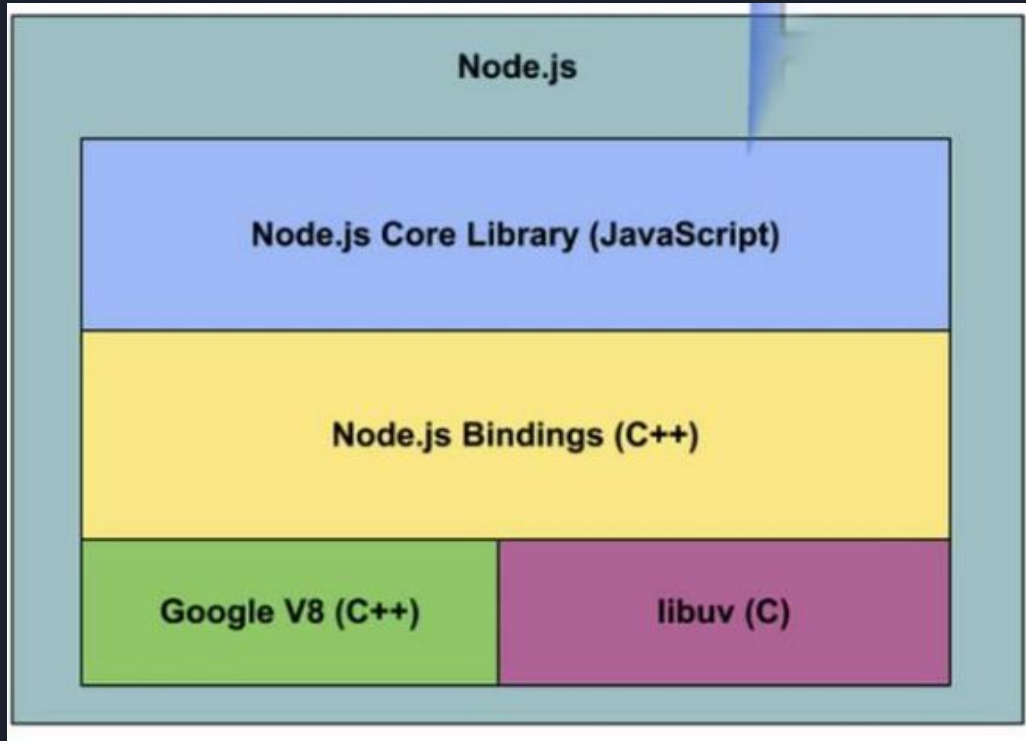
What is Node Js actually ?

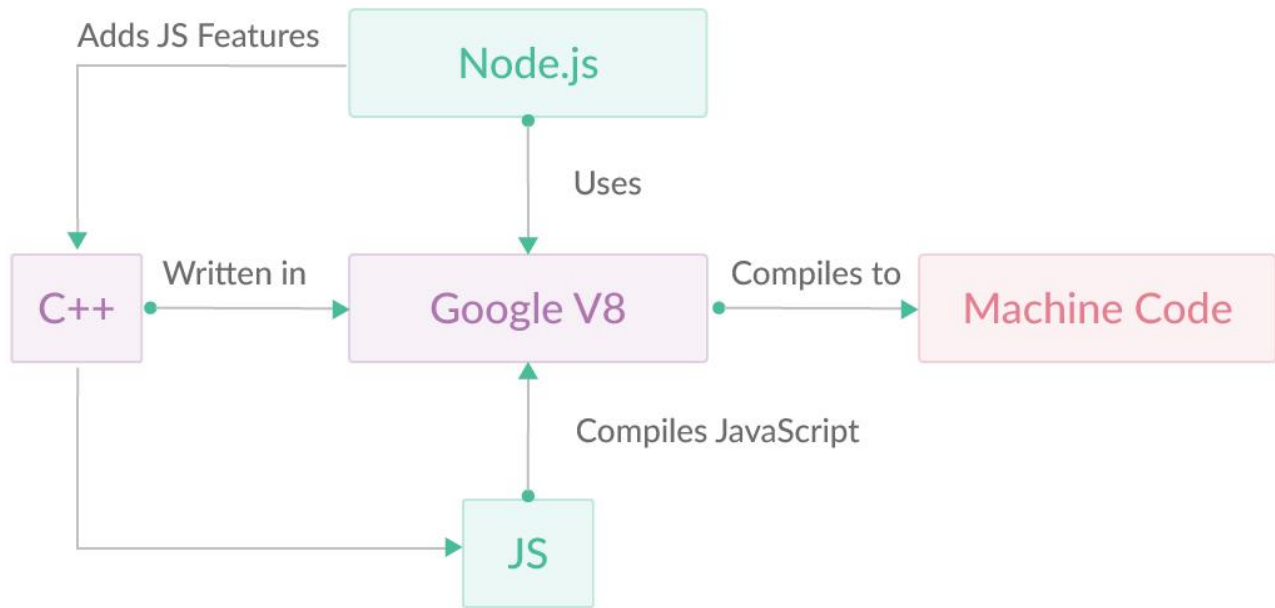
- Node JS is not a language it is a runtime environment, just like how we run JS in Browser Environment.
- It is a tool, we use it build any projects but code in JS.
- Under the hood of any Browser, there is a engine that runs JS code, for example :
 - V8 engine for Google Chrome
 - SpiderMonkey for Mozilla
 - JSCore for Safari
 - Chakra for Internet Explorer and now V8 for Edge
- Similarly Node uses V8 engine to run JS code outside of the Browser, additionally it uses few libraries and C++ bindings for internal process communication.


References:

1. https://en.wikipedia.org/wiki/JavaScript_engine

Node JS Engine







We need to understand important features of Node JS so that we can get the context of things work with Node.





Node JS Features & Advantages

1. Non - blocking I/O
2. Event driven
3. Asynchronous
4. Fast and Scalable
5. Community Support
6. Single Language Code Base
7. Open Source
8. Integration with any JS stacks



Project Environment Setup

1. Install NodeJs from here - <https://nodejs.org/en/download/> (LTS Version only, current version is 14)
2. Install VS Code from here - <https://code.visualstudio.com/download>
3. Install Git Bash (if needed)
 - a. Git is natively supported in Ubuntu.
 - b. You need Git Bash to install in Windows, download from here - <https://git-scm.com/download/win>
 - c. While installing select the option to add Git bash to as default cmd, so the Git will be accessible in cmd.



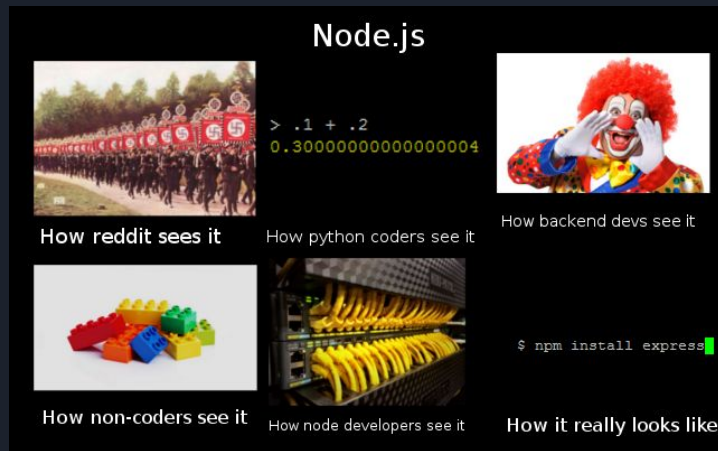
How to start node project and the Basics

Type in CMD:

1. “node -v “ - this will verify installed version of the node.
 2. “git -v ” - this will verify if Git is installed and also accessible in cmd
-
1. Go to the folder of the project (name the folder anything you want)
 2. >>> npm init
 3. This will start the project for Node Js and it will ask few basic questions about the project, type whatever you want to.

What is npm ?

- NPM is Node Package Manager.
- It is online registry from where you can get information regarding libraries, third party libraries, frameworks and can download those packages.
- With the help of “npm install package_name” command you can install any existing library from npmjs registry.
- NPM to Node JS is what PIP is to Python.



Creating the First App

Command Line Calculator





Lesson 1 : Hello World

- First programm:
 - `console.log("Hello World!!!");`
- How to Run:
 - “node helloworld.js”



Lesson 2 : Function

```
function greetings(message, name) {  
    console.log(message + " " + name);  
}
```

```
greetings("Hello", "people");
```



Lesson 3 : process and command line arguments

- A global object available to Node JS applications without importing it.
- It has information about the current node process and has few function to provide control over it.
- Code:
 - `console.log(process);`
 - Outputs information about current node process
 - `console.log(process.env);`
 - Outputs all the environments variables stored in the system



How to command line input

- Code:
 - `console.log(process.argv);`
 - Outputs all the argument provided against the command line in form of an array.

- How to run:


```
>>> C:\Users\user\Desktop\sttp_node>node command_calculator/main.js  
--name=john --surname=doe
```

```
[  
  'C:\\Program Files\\nodejs\\node.exe',  
  'C:\\Users\\user\\Desktop\\sttp_node\\command_calculator\\main.js',  
  '--name=john',  
  '--surname=doe'  
]
```



Removing "--" and "=" from the argument strings

- `string.split("--")` → will split the string into array by separating by the string / character specified.
- Code:
 - `--name=john".split("--")`
 - Output will be [`'`, `'name=john'`] as string is split into array at breakpoint of `--`.
- Now we can join this output array into string, so that original string, it does not contain `--` anymore.
- `array.join("")` → this will join (concat) all the values in array to form a final string.
- Code:
 - `[', 'name=john'].join("")`
 - Output will be `"name=john"`, one single string.

- 
- First value is related to the path of the node environment.
 - Second value is related to the path of JS file being executed.
 - Result of the values are nothing but the command line arguments entered by the user.
 - We only want values after second argument.
 - Code:
 - ```
console.log(process.argv.slice(2));
```
    - Output, the slice function returns new array starting from index 2 (inclusive).
  - If you check the output then you'll see we have extras like "--" & "=" in "--name=john", we only want values that is "name" and "john", so we need to remove it.



# App code till node

```
const argumentsArray = process.argv.slice(2);

// loop through array of arguments
for(let index = 0; index < argumentsArray.length; index++) {
 let splitArray = argumentsArray[index].split("--");
 let joinedString = splitArray.join("")
 console.log(joinedString)
}
```

- Output:

node command\_calculator/main.js --name=john --surname=doe

name=john

surname=doe

# Now split the arguments in key-value pair objects

- Key-value pair example :

```
let someObject = {
 key1: "value1",
 key2: "value2"
}
```

- Since we have "name=john", we need to split into 'name as key' and 'john as value' and removing "=" completely.
- For that we will again use split function and output will produce array containing key and value.
- Code :
  - `let keyValueArray = joinedString.split("=");`
  - Output will be

[ 'name', 'john' ]

[ 'surname', 'doe' ]



# App code till now

```
let arguments = {};
const argumentsArray = process.argv.slice(2);

// loop through array of arguments
for(let index = 0; index < argumentsArray.length; index++) {
 let splitArray = argumentsArray[index].split("--");
 let joinedString = splitArray.join("");
 let keyValueArray = joinedString.split("=");
 arguments[keyValueArray[0]] = keyValueArray[1];
}

console.log(arguments)
```

Output:

```
node command_calculator/main.js --name=john --surname=doe
```

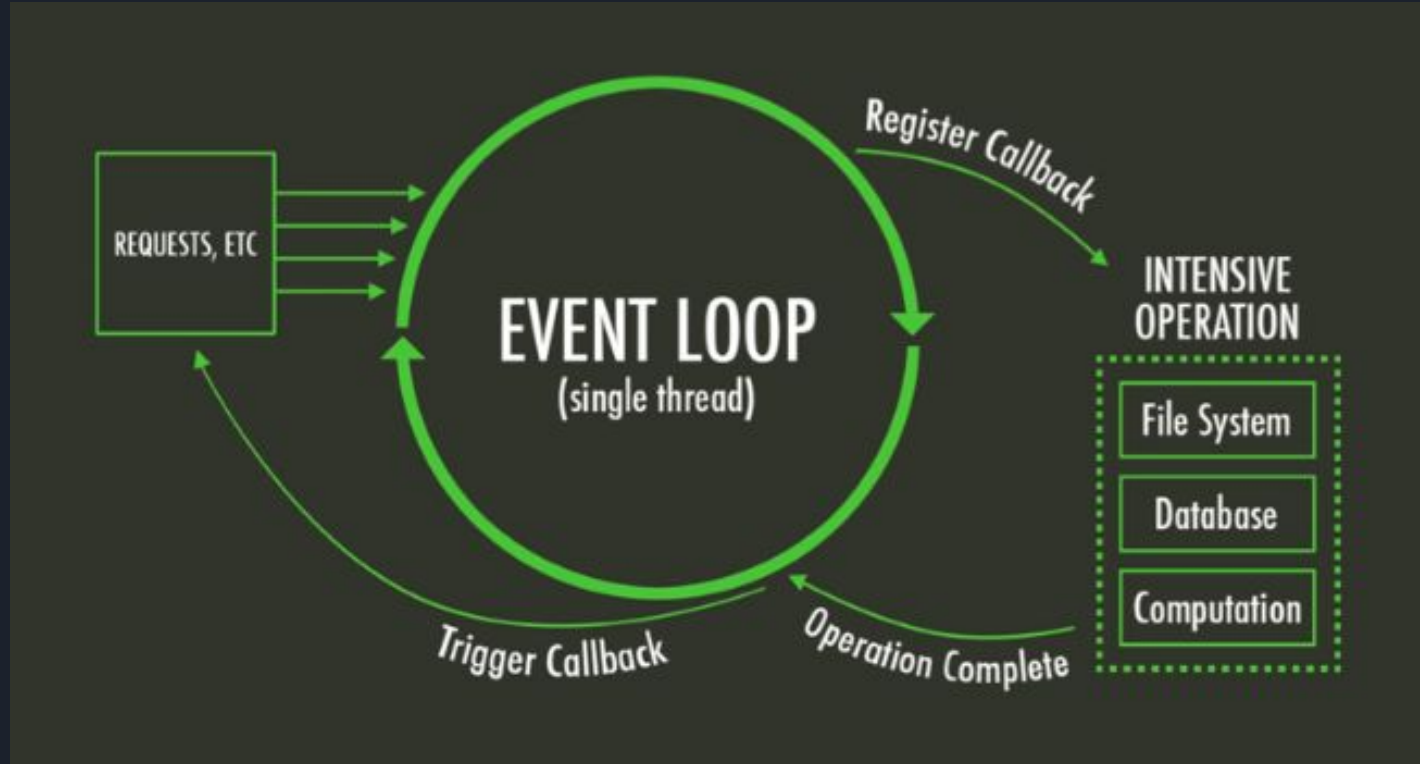
```
{ name: 'john', surname: 'doe' }
```



Lesson 3 final code → git branch = “lesson-3”

Link: [https://github.com/10xtarun/sttp\\_node\\_2021/tree/lesson-3](https://github.com/10xtarun/sttp_node_2021/tree/lesson-3)

# Lesson 4 : Asynchronous Programming



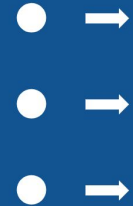
# Synchronous Vs Asynchronous

- In sync operations, the process completes the task then it hands over back the main control, till then no other task can be performed.
- In async operations, the process is handed over to worker threads and they completes the task and returns the output to the process back.
- In this while the main process is free and hence it can perform other tasks.

## Synchronous

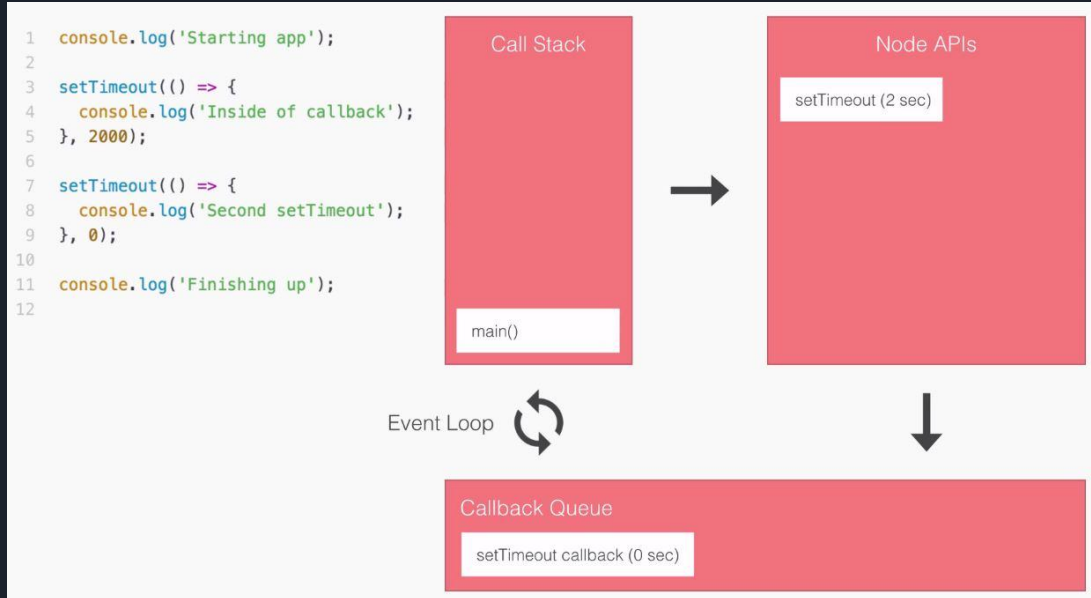


## Async



# Callbacks

- Callbacks are function which are passed as arguments in other functions and these callback is trigger when the task/request is completed.
- A triggered callback will contain, error if any, data and other options if specified.
- This callback returns the data to main process.



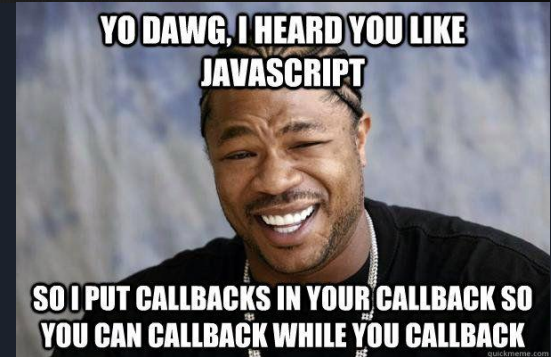


# Problem: “Callback hell”

Drawbacks of using callback:

- Even though callbacks provide flexibility to handle async situation but it is a hard concept to understand.
- Using callbacks can lead code unreadability, it becomes difficult to keep track on things.

```
1 var asyncJavaScript = function(err, callback) {
2 callback(function(err, callback) {
3 callback(function(err, callback) {
4 callback(function(err, callback) {
5 callback(function(err, callback) {
6 callback(function(err, callback) {
7 callback(function(err, callback) {
8 callback(function(err, callback) {
9 console.error('CALLBACK HELL');
10 });
11 });
12 });
13 });
14 });
15 });
16 });
17 });
```



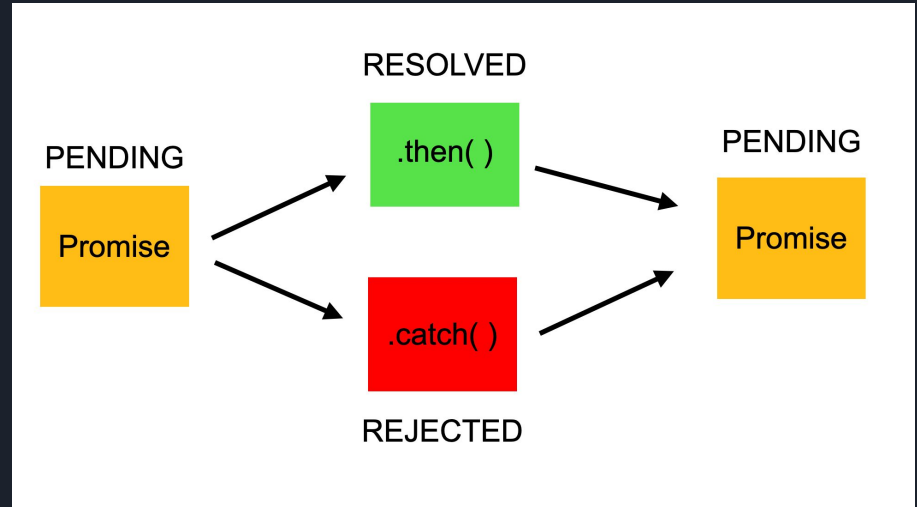
# Async Programming: Using Promises

- All async code/logic can be converted to Promises.
- A promise has two outcomes: Rejected or Resolved.
- Rejected is triggered when Promise is not fulfilled (async task fails or generates error)
- Resolved is triggered when Promise is fulfilled.



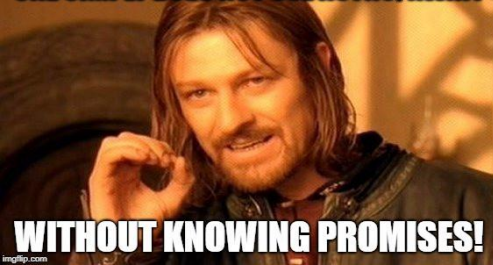
# Async Programming: Using Promises

- Resolved promises are handled with then block which is again a promise.
- Similarly rejected promises are handled in catch block.
- Above both can be returned when if fill condition is enough for success or failure.
- Or chain is continued.



# Async Programming: Using async - await

ONE SIMPLY DOES NOT USE ASYNC/AWAIT



Use async keyword  
before function  
definition

await keyword that works only  
inside async functions

Fetch keyword for  
network calls (i.e HTTP  
Client)

```
async function getUsers(url) {
 let response = await fetch(url);
 let data = await response.json();
 console.log(data);
 return data;
}
```

If promise fulfills, you  
will get the values  
back else rejected  
value is thrown

When you await, a  
promise function  
paused in non blocking  
way until it settles

# Thank You

- Tarun Singh
- 10xtarun@gmail.com



All images and links are referenced from google.com